Lock in Feedback in Sequential Experiments

Maurits Kaptein and Davide Iannuzzi[*]

Abstract: We often encounter situations in which an experimenter wants to find, by sequential experimentation, $x_{max} = \arg\max_x f(x)$, where $f(x)$ is a (possibly unknown) function of a well controllable variable x. Taking inspiration from physics and engineering, we have designed a new method to address this problem. In this paper, we first introduce the method in continuous time, and then present two algorithms for use in sequential experiments. Through a series of simulation studies, we show that the method is effective for finding maxima of unknown functions by experimentation, even when the maximum of the functions drifts or when the signal to noise ratio is low.

1. Introduction

When designing an experiment where a given parameter must be kept constant throughout the entire duration of the measurement, physicists and engineers often rely on feedback techniques that, in real time, can properly re-adjust the configuration of the experiment to compensate for unexpected drifts (Scofield, 1994). Fig. 1 illustrates, for instance, a well-established approach that is used to maintain a variable x always locked at the value that maximizes the value of another variable y, which is some function – possibly with large noise – of x. The algorithm behind this approach, which will be described more in depth later in the text, is based on the following steps:

1. Fix a central value x_0 of the variable x;
2. Add an oscillation of amplitude A at a fixed angular frequency ω: $x = x_0 + A_x \cos(\omega t)$.
3. Measure the amplitude of the oscillations that the variable y has, in response of the oscillation of the variable x, at the same angular frequency ω, and further measure whether the oscillation are in phase or out of phase;
4. Set a new value of x_0, adding (if the oscillation of y are in phase with the oscillation of x) or subtracting (if the oscillation of y are out of phase with respect to the oscillation of x) a value proportional to the value measured in step 4: $x_{0,new} = x_0 \pm \gamma A_y$, where γ is a constant. Iterate steps 2 to 4 for the whole duration of the experiment.

The above described feedback loop pushes the value of x_0 closer and closer to the value x_{max} that maximizes y. As x_0 approaches x_{max}, the oscillations

---

in y become smaller and smaller, moving x_0 in a series of steps of decreasing size. Finally, when $x_0 = x_{max}$, the variable y ceases to oscillate at frequency ω: because of this x_0 can stay locked in on x_{max}. However, if the curve suddenly shifts to another position (e.g., if the relationship between x and y changes, a phenomenon referred to as concept drift (Gaber et al., 2005; Anagnostopoulos et al., 2012)), the ω component of y becomes different from zero again, forcing the feedback loop to move the value of x_0 towards the new value of x_{max}. Hence, the feedback loop enables one to hold on to the value of x sequentially that maximizes the value of y.

Interestingly, the feedback loop described above can work well even if the variable y is affected by a high degree of noise. To extract the signal at frequency ω, in fact, one can make use of a commercial instrument called *lock-in amplifier*, which rejects all the components of the signals that do not beat at the frequency of interest. The algorithm used by a lock-in amplifier can of course be applied to digital (discrete timepoints) data as well. It is thus worth asking whether the approach adopted in a lock-in amplifier may be used in other contexts where, in the presence of a highly noisy set of data, one wants to maintain one variable locked to the value that maximizes the value of another.
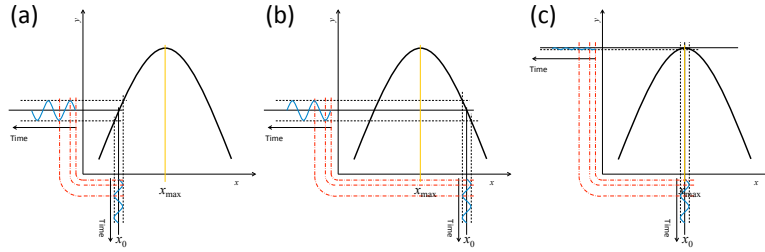

FIG 1. *Illustration of the lock-in principle used in physics and engineering to maintain a bring and maintain an independent, controllable variable x onto the value x_{max} for which a dependent variable y is maximized. The value of x is oscillated sinusoidally around a central value x_0. (a): If $x_0 < x_{max}$, y oscillates at the same frequency as x, in phase (i.e., a maximum value of x corresponds to a maximum value of y). (b): If $x_0 > x_{max}$, y oscillates again at the same frequency as x, but with opposite phase (i.e., a maximum value of x corresponds to a minimum value of y). (c): If $x_0 = x_{max}$, y ceases to oscillate at the frequency of x, but starts to oscillate at a doubled frequency. Lock-in amplifiers can detect the amplitude and the phase of the oscillation at a reference frequency, and, therefore, indicate whether x is smaller, larger, or equal to x_0.*

Tantalized by this opportunity, we propose here to use lock-in feedback (LiF) algorithms for the optimization of the price in (e.g.,) a rebate action. The idea is to present each customer a different price, which is changed sinusoidally around a central value, causing the revenue to oscillate at the same frequency. As the customers take their purchasing decision, a lock-in algorithm monitors the oscillations of the revenue at the price oscillation frequency. Like in the feedback loop described above, the central value of the price is continuously adjusted until the revenue ceases to oscillate at that frequency. At this point, in fact, the revenue is maximum (price elasticity = 1). If an unexpected event moves the

price elasticity curve, the algorithm will automatically push the central price towards the new maximizing value.

Next to product pricing of rebate actions, many more examples could be conceived in the social sciences:

- In economics, firms might be able to manipulate the price x of an offering and subsequently observe their revenue y. Here a firm seeks to find the value of x that maximizes y (for examples see Kung et al., 2002; Jiang et al., 2011).
- In industry, the outcome y of a business process might depend on the amount of some raw material x used in the process.
- In communication research, a communication professional might seek to find the length of an email message x that leads to the highest number of clicks y on a link in that message (Ansari and Mela, 2003).
- In medicine, a physician seeks to find the optimal dose x of a medicine to maximize the health outcome y of her patients (see, e.g., Sapareto and Dewey, 1984; Marschner, 2007).
- In education, scholars might seek to select learning tasks which are quantified by their difficulty x, that have the highest effect on learning y of their pupils.

In the above cases the functional form of $f(x)$ is often not known, the outcome y is observed with noise, and likely the treatment values that maximize the outcome are subject to concept drift (Gaber et al., 2005; Anagnostopoulos et al., 2012) (thus, they change over time). Here we present a method to find x_{max} which does not require an explicit specification of $f(x)$ or its derivatives, performs well in the face of noise, and is robust to concept drift.

To prove the merits of LiF in such cases, we have performed an extensive numerical exercise that simulates the performance of LiF in a diverse range of situations, including ones where the observed signal is merely the choice of a consumer to yes or no adopt a product for a given (rebate) price; a scenario directly in line with the pricing challenges as identified above. We show that, in the presence of the noise induced by the variance of the willingness to pay across the population of the customers entering the shop, our lock-in algorithm allows the seller to both determine and maintain the price that optimizes the revenue of the shop. Furthermore, we demonstrate that if the price elasticity curve changes, the algorithm can detect the direction of the change and converge again to the optimal price.

It has to be noted that it is a well-known and well-studied challenge to find optimal (according to some specified criterion) treatment values in (sequential) experiments. This challenge is acknowledged in many branches of science and engineering (see, e.g., Allen et al., 2003; Bardsley et al., 1996; Kuck et al., 2006). An often researched topic is that of design optimization (DO), in which experimental designs are identified that lead to the smallest possible variances in the estimated model parameters (Burnetas and Katehakis, 1996; McClelland, 1997). More recently, an interest in adaptive design optimization (ADO) methods (Myung and Pitt, 2009b; Myung et al., 2013) and sequential experimentation

methods has emerged: researchers are looking for effective ways to sequentially determine optimal treatment values in experiments as the experimental data is being collected (Zhang and Lee, 2010). Notably, work on Multi Armed Bandit (MAB) problems (e.g., Lai, 1987; Whittle, 1980; Scott, 2010; Bubeck et al., 2011a; Yue et al., 2012) and stochastic optimization (e.g., Agarwal et al., 2011) has led to efficient sequential sampling schemes for various experimental designs and optimization criteria.

This paper however introduces a novel sequential sampling scheme for a specific sequential design problem: we examine the problem in which the treatment values are continuous (e.g., with x being $\in \mathbb{R}$) and the researcher seeks a treatment value x_{max} at which the observed outcome y—which, at least in part, depends on x—obtains its maximal value. Thus we examine the situation in which an experimenter wants to find, by sequential experimentation, $x_{max} = \arg\max_x f(x)$, where $f(x)$ is a (possibly unknown) function of a well controllable variable x and is likely observed with noise. We focus on the simple case where x is a scalar. In the remainder of the paper, we index sequential trials by $t \in \{1, \ldots, \mathcal{T}\}$. Our ultimate aim is to describe an experimental method for manipulating x_t (in discrete time) to find, sequentially, the value of x that maximizes y.

The current manuscript is structured as follows: first, we briefly review the literature on DO, MAB problems, and stochastic optimization to position our method. Next, we discuss LiF as a solution to the treatment optimization problem considered in this paper. LiF is based on a solution that is routinely implemented in physics and engineering applications which relies on the idea of systematically changing the value of the treatment in time via so-called *lock-in amplifier techniques* (Scofield, 1994). We introduce its basic principles in continuous time. Subsequently, we present two algorithms to use LiF in sequential experiments. We then, by simulation, compare the two algorithms, and examine the performance of LiF in several scenario's of signal-to-noise ratio and in situations of concept drift. Furthermore, we examine the use of LiF in cases in which the observable outcome is discrete; which is for example the case in the optimization of prices as described above. Finally, we examine the empirical *regret* – the search cost of the algorithm compared to an algorithm which has full information – of the proposed procedure and compare it to a standard solution in the MAB literature (Berry and Fristedt, 1985).

1.1. Treatment optimization methods

The problem of finding x_{max} is treated in a number of branches in the experimental design and machine learning literature. The problem can be approached as an optimal design problem, in which the main aim is to design an experiment that efficiently provides us with information regarding $f(\vec{x})$ (see, e.g., O'Brien and Funk, 2003; Myung and Pitt, 2009a). Often, in the DO literature, experiments are treated statically, and the functional form of the data generating function is assumed known: the remaining question is to determine the optimal

treatments given a fixed size of the experiment and an assumed relationship to precisely estimate the parameters of interest.

Recently, (Myung et al., 2013) introduced an advanced method of DO into the psychology literature called Adaptive Design Optimization (ADO). The aim of ADO is to create adaptive experiments which are optimized to distinguish between competing explanations of the data (Myung and Pitt, 2009b). However, in this literature the main aim is to find treatment values to efficiently estimate parameters given a number of model assumptions. Instead, our focus is on efficiently finding treatment values which maximize some observable outcome of the experiment.

Sequentially finding optimal treatments, where optimal is defined in terms of observed outcomes, is explicitly studied in the MAB literature (Berry and Fristedt, 1985). In this problem specification researchers consider policies $\mathcal{P}$ which describe how to select actions $a \in \mathcal{A}$ (the treatment values) at different times t where the aim is to maximize the cumulative reward $R(t) = \sum_{t=1}^{T} r_i$ (Bubeck et al., 2011a). The reward is assumed to be a function, possibly with noise, of the actions. Many specifications of the MAB problem exists: researchers have considered independent treatments (the traditional k-armed bandit problem (Whittle, 1980)), related treatments, continuous treatments, etc. (Audibert et al., 2009; Bubeck et al., 2011b). The MAB problem, and its generalization, the contextual MAB problem (Li et al., 2010; Beygelzimer et al., 2011) present an active area of research in the machine learning literature.

The literature on stochastic optimization with bandit feedback (Agarwal et al., 2011, 2010) considers the problem of finding the optimal value of continuous treatments (Flaxman et al., 2005). Of special interest for the current proposal are derivative-free (or gradient-free) methods in which the gradient of the function (which is of use for e.g., (stochastic) gradient descent method) is assumed unknown and is itself approximated during the sequential experiment (Shamir, 2012). In this paper we present a derivative free method to perform stochastic optimization with bandit feedback. The presented method is well-suited for practical use in sequential experiments due to its ease of implementation: in the current paper we provide several algorithms for performing the optimization in real-life settings. Before presenting our novel sequential approach to solving the continuous treatment optimization problem, we first introduce its theoretical background assuming that the treatment does not vary in discrete sequential steps, but rather can be varied continuously (in continuous time).

2. Finding the maximum of a curve with a lock-in algorithm

In this section we detail the basic principles behind LiF assuming continuous time in which x can be manipulated. Let's assume that y is a continuous function f of x: $y = f(x)$. Let's further assume that x oscillates with time according to:

$$x(t) = x_0 + A \cos(\omega t) \tag{1}$$

where ω is the angular frequency of the oscillation, x_0 its central value, and A its amplitude. For relatively small values of A, Taylor expanding $f(x)$ around

x_0 to the second order, one obtains:

$$y(x(t)) = f(x_0) + (x_0 + A\cos(\omega t) - x_0)\left(\left.\frac{\partial f}{\partial x}\right|_{x=x_0}\right)$$
$$+ \frac{1}{2}(x_0 + A\cos(\omega t) - x_0)^2\left(\left.\frac{\partial^2 f}{\partial x^2}\right|_{x=x_0}\right) \tag{2}$$

which can be simplified to:

$$y(x(t)) = k + A\cos(\omega t)\left(\left.\frac{\partial f}{\partial x}\right|_{x=x_0}\right)$$
$$+ \frac{1}{4}A^2\cos(2\omega t)\left(\left.\frac{\partial^2 f}{\partial x^2}\right|_{x=x_0}\right) \tag{3}$$

where $k = f(x_0) + 1/4A^2\left(\left.\partial^2 f/\partial x^2\right|_{x=x_0}\right)$. It is thus evident that, for small oscillations, y becomes the sum of three terms: a constant term, a term oscillating at angular frequency ω, and a term oscillating at angular frequency 2ω.

Suppose we ourselves can actively manipulate x and measure y, and that f is continuous and only has one maximum and no minimum.[1] Further suppose that one is interested to find the value $\arg\max_x y = f(x)$ which we denote with x_{max}, and that our measurements of y contain noise

$$y(t) = f(x(t)) + \epsilon_t \tag{4}$$

where ϵ denotes the noise and $\epsilon \sim \pi()$ where π is some probability density function and $\mathbb{E}[\epsilon|x] = 0$.

Following the scheme used in physical lock-in amplifiers (see, e.g., Scofield, 1994), we multiply the observed y variable by $\cos(\omega t)$. Using eq. 3 and eq. 4, one obtains:

$$y_\omega(t) = \cos(\omega t)\left[k + A\cos(\omega t)\left(\left.\frac{\partial f}{\partial x}\right|_{x=x_0}\right)\right.$$
$$+ \frac{1}{4}A^2\cos(2\omega t)\left(\left.\frac{\partial^2 f}{\partial x^2}\right|_{x=x_0}\right) + \epsilon\bigg] \tag{5}$$

where y_ω is the value of y after it has been multiplied by $\cos(\omega t)$. Eq. 5 can be written more compactly as:

$$y_\omega = \frac{A}{2}\left(\left.\frac{\partial f}{\partial x}\right|_{x=x_0}\right)$$
$$+ k_\omega\cos(\omega t) + k_{2\omega}\cos(2\omega t)$$
$$+ k_{3\omega}\cos(3\omega t) + \epsilon\cos(\omega t) \tag{6}$$

---

[1]For simplicity of exposure we only consider these well-behaved functions in this paper.

where

$$k_\omega = k + A^2/8 \left(\partial^2 f/\partial x^2 \big|_{x=x_0} \right) \tag{7}$$

$$k_{2\omega} = A/2 \left(\partial^2 f/\partial x^2 \big|_{x=x_0} \right) \tag{8}$$

$$k_{3\omega} = A^2/8 \left(\partial^2 f/\partial x^2 \big|_{x=x_0} \right) . \tag{9}$$

Integrating y_ω over a time $T = \frac{2\pi N}{\omega}$, where N is a positive integer and T denotes the time needed to integrate N full oscillations, one obtains:

$$y_\omega^* = \frac{TA}{2} \left(\frac{\partial f}{\partial x} \bigg|_{x=x_0} \right) + \int_0^T \epsilon \cos{(\omega t)}\, dt \tag{10}$$

Depending on the noise level, one can tailor the integration time, T, in such a way to reduce the second addendum of the right hand of eq. 10 to negligible levels, effectively averaging out the noise in the measurements. Under those circumstances, y_ω^* provides a direct measurement of the value of the first derivative of f at $x = x_0$.

The above method thus yields quantitative information regarding the first derivative of f at $x = x_0$, providing, in this way, a logical update strategy of x_0: if $y_\omega^* < 0$, then x_0 is larger than the value of x that maximizes f; likewise, if $y_\omega^* > 0$, x_0 is smaller than the value of x that maximizes f. Thus, based on the oscillation observed in y_ω we are now able to move x_0 closer to $x = \arg\max_x f(x)$ using an update rule $x_0 := x_0 + \gamma y_\omega^*$ where γ quantifies the learn rate of the procedure. Hence, we can setup a feedback loop that allows us to keep x_0 close to x_{max}, even if $f(x)$ changes over time.

Note that, multiplying y by $\cos 2\omega t$ and using a similar approach as the one described above to extract the amplitude of the oscillation of y at frequency 2ω, one would be able to measure the second derivative of the function f at $x = x_0$. This property can be useful when, for instance, $f(x)$ is known to be an exact parabola to not only derive the direction of the step towards the maximum, but to work out the exact step size (see Appendix 9).

3. Algorithm for LiF in discrete time

In practical terms, measurements can never run in continuous mode. Therefore, we now present an algorithm for LiF in discrete time. To simplify notation, we will index sequential measurements by y_t where $t = 1, \ldots, t = \mathcal{T}$ where $\mathcal{T}$ denotes the length—possibly infinite—of the experiment that is ran to find $\arg\max_x f(x)$.

In discrete time we can use the same procedure as above in which we start with x_0, and for each sample oscillate around x_0 with a known frequency ω and known amplitude A:

$$x_t = x_0 + A \cos \omega t \tag{11}$$

which will result in measurements given by

$$y_t = f(x_0 + A \cos \omega t) + \epsilon_t \tag{12}$$

On the basis of the arguments reported above, we can now implement a feedback loop that iteratively adjusts the value of x_0 until x reaches x_{max}. After that, if the function f changes, the loop can follow the value of x to the new maximizing position and thus stay "locked". The procedure is similar to that given in Equation 6 and 10, where we first multiply the outcome y_t by $\cos(\omega t)$ and subsequently integrate out the noise term (summing in the discrete case). In the following sections we present two possible implementations for LiF in discrete time for use in sequential experiments.

3.1. LiF-I: Batch updates of x_0

Our first implementation of LiF (denoted LiF-I) is presented in Algorithm 1. In this implementation we summate observations y_t, which we multiply by $\cos(\omega t)$, for a batch period of length T, after which we update x_0. Variable y_ω^Σ contains a running sum of $y_t \cos \omega t$ over t that is used for the integration.

Algorithm 1 Implementation of LiF-I for single variable maximization in data stream using a batch approach.

Require: x_0, A, T, γ, $y_\omega^\Sigma = 0$
  $\omega = \frac{2\pi}{T}$
  for $t = 1, \ldots, \mathcal{T}$ **do**
    $x_t = x_0 + A \cos \omega t$
    $y_t = f(x_0 + A \cos \omega t) + \epsilon_t$
    $y_\omega^\Sigma = y_\omega^\Sigma + y_t \cos \omega t$
    if $(t \mod T == 0)$ **then**
      $y_\omega^* = y_\omega^\Sigma / T$
      $x_0 = x_0 + \gamma y_\omega^*$
      $y_\omega^\Sigma = 0$
    end if
  end for

The tuning parameters for LiF-I, which should be set by the experimenter, are x_0, A, T, γ. Here below we describe some general criteria the choice may be based on:

- It is advised to set x_0 as close as possible to x_{max}. The choice can only be based on the available information on f. The more accurate the information, the closer the initial x_0 to x_{max}, the faster the convergence of the loop to x_{max}.
- The amplitude A affects the costs of the search procedure, because a large A implies querying a large range of x values with (possibly) low resulting y values. However, A also influence the learning speed: a very small A leads to small updates steps, while a large value of A might lead to a value of γy_ω^* that "overshoots" x_{max}.

- The integration time T affects the variability of the update of x_0, with larger integration times leading to a smoother update but slower convergence.
- The learn-rate $\gamma < 1$ determines the step size at each update of x_0. This can be interpreted, and tuned, akin learn-rates in, for instance, stochastic gradient descent methods (Poggio et al., 2011).

3.2. LiF-II: Continuous updates of x_0

For some applications the batch updates of x_0 – as implied by the continuous time analysis and defined in Algorithm 1 – might not be feasible. Algorithm 2 presents a modified version of LiF (denoted LiF-II) in which x_0 is updated every observation. LiF-II starts by filling up a buffer of length T which we denote by the vector $\vec{y}_\omega = \{NA_1, \ldots, NA_T\}$, after which each observation leads to an update of x_0. In the algorithm description the values $y_{t-T}, \ldots, y_t$ are stored in the vector $\vec{y}_\omega$. By defining the learn rate as $\frac{\gamma}{T}$ the tuning parameters in LiF-II are the same as those discussed for LiF-I.

Algorithm 2 Implementation of LiF-II for single variable maximization using continuous updates.

Require: x_0, A, T, γ, $\vec{y}_\omega = \{NA_1, \ldots, NA_T\}$
  $\omega = \frac{2\pi}{T}$
  for $t = 1, \ldots, \mathcal{T}$ **do**
    $x_t = x_0 + A\cos\omega t$
    $y_t = f(x_0 + A\cos\omega t) + \epsilon_t$
    $\vec{y}_\omega = \texttt{push}(\vec{y}_\omega, y_t\cos\omega t)$
    if $(t > T)$ **then**
      $y_\omega^* = (\sum \vec{y}_\omega)/T$
      $x_0 = x_0 + \frac{\gamma}{T}y_\omega^*$
    end if
  end for

---

4. Simulation study 1: Comparison of Batched and streaming LiF and examination of tuning parameters

In this section we study, by simulation, the differences between LiF-I and LiF-II, and the effects of the tuning parameters A, T, and γ in a situation in which $y = f(x)$ is measured without noise.

Figure 2 presents the performance of both LiF-I and LiF-II for data generated using

$$f(x) = -2(x-5)^2 + \epsilon \tag{13}$$

where $\epsilon \sim \mathcal{N}(0,0)$ and obviously $x_{max} = 5$. The figure displays the performance of LiF for $\mathcal{T} = 10000$ using the following tuning parameter settings
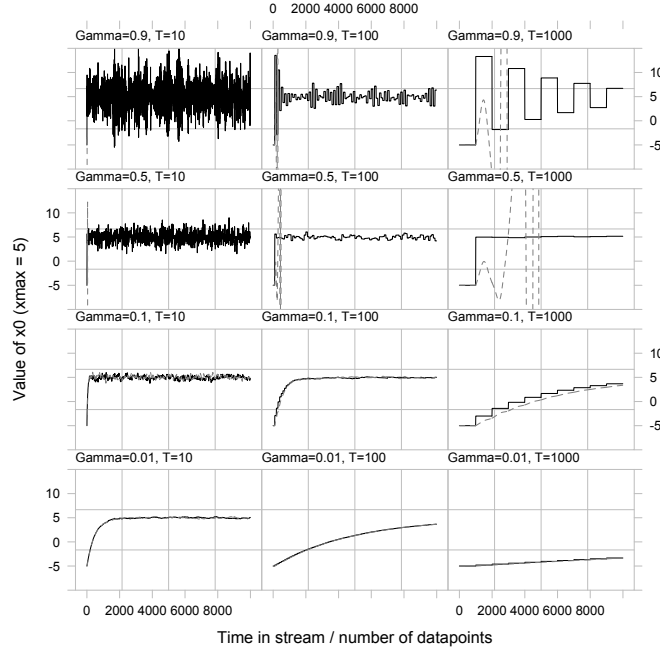
- $x_0 = -5$.

FIG 2. *Examination of the effect of LiF tuning parameters γ and T for A=1. Displayed are the results for LiF-I (black solid line) and LiF-II (gray dotted line)*

- $T \in \{10, 100, 100\}$
- $A = 1$
- $\gamma \in \{.01, .1, .5, .9\}$

The rows of Figure 2 (top to bottom) present decreasing values of γ, while the columns (left to right) present increasing values of T. We fix $A = 1$. Each panel presents the value of x_0 during the data stream as selected using LiF-I (black solid line) and LiF-II (gray dotted line). It is clear that LiF can "overshoot" the maximum for values of γ that are too high (top two rows). This happens for both LiF-I and LiF-II, although LiF-I seems more robust. For small values of γ the performance of the algorithms is very similar, and increases in the integration window T merely smooth the updating procedure.

In Figure 3 the results are plotted for the same setup, but this time we vary $A \in \{.1, 1, 2, 10\}$, while we fix $\gamma = .1$. Here it is clear that for large values of A LiF-I has a tendency to become unstable (see top rows), while the streaming LiF-II is much more robust for erroneous selection of A. Very small choices for the amplitude A lead to very slow updates of x_0 in both cases. Again, increased in T merely smooth the process. The simulations give an impression of the importance of the tuning parameters x_0, A, T, γ and their relationships. In the remainder of this paper we will focus on the evaluation – through simulation –
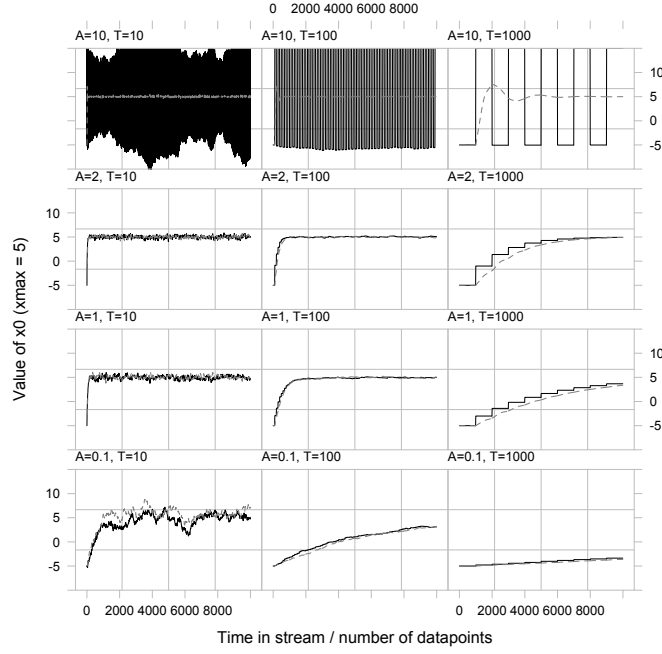
FIG 3. *Examination of the effect of tuning parameters A and T for γ = .1. Displayed are the results for LiF-I (black solid line) and LiF-II (gray dotted line)*

of the performance of LiF-II in cases of noise and concept drift.

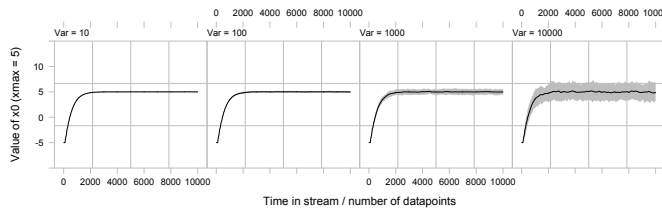## 5. Simulation study 2: Effects of noise

FIG 4. *Examination of the effect of different levels of noise $\sigma^2 \in \{10, 100, 1000, 10000\}$. Note that LiF performs very well also in the presence of noise (see text for more details).*

To examine the impact of (measurement) noise on the performance of LiF-II we repeat the simulations as described in Simulation Study 1 using the data generating model described by Equation 13 with $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and $\sigma^2 \in$

$\{10, 100, 1000, 10000\}$. We choose tuning parameters: $x_0 = -5$, $A = 1$, $T = 100$, $\gamma = .1$. Contrary to the simulations presented in Section 4 we now repeat the procedure $m = 100$ times: Figure 4 presents the average x_0 over the 100 simulation runs as well as the 95% confidence bounds. From Figure 4 it is clear that LiF-II performs very well in the face of noise.

6. Simulation study 3: Performance of LiF-II in cases of concept drift

One of the advantages of Lock in Feedback as opposed to other methods of finding x_{max} is the fact that LiF can also be used to find a maximum of a function in cases of concept drift (Gaber et al., 2005): even when $f(x)$ changes over time, LiF provides a method to keep the value of the treatment x close to x_{max}.

To illustrate this latter advantage of LiF-II we setup a simulation using the following data generating model:

$$f(x, t) = -2((x - .0025t) - 5)^2 + \epsilon \tag{14}$$

where the $(x - .0025t)$ term ensures that during the stream running from $t = 0$ to $t = 10^4 = \mathcal{T}$ the value of x_{max} moves from 5 to 30. We choose $x_0 = -20$ (note the different starting position compared to the previous simulations), $A = 1$, $T = 100$, $\gamma = .1$ and $\sigma^2 = 10$. We investigate the performance of LiF-II in this case of concept drift.

Figure 5 presents in the top panel $y = f(x, t)$ for distinct values of $t \in \{0, 1000, \ldots, 10000\}$ in different shades of grey. The concept drift is illustrated by the different locations of the parabola. Superimposed in blue is the value of x_0 as selected by LiF-II. In the bottom panel the value of x_0 as a function of the length of the stream is presented. It is clear that LiF-II quickly finds x_{max} and follows the maximum as it moves during the stream.

7. Simulation study 4: Dichotomous observations

In the introduction we described as a use case of our proposed method the optimization of sales prices to maximize the revenue. This specific case presents a novel problem since the dependent variable y, encoding the purchase decision of a customer after a price has been pitched is dichotomous, and the actual outcome of interest—if the firm aims to maximize its revenue—is a function of the observable and the manipulated variable $r(t) = \sum_{i=1}^{t} y_i x_i$. Since $y_i \in \{0, 1\}$, the signal $r(t)$ used as an optimization criteria contains a different type of noise; while the expected value $Pr(y = 1|x) \times x$ of an offer could be approximated, the data itself contains non-zero values only when the decision is made to purchase a product.

To empirically examine the performance of LiF in such a setting we setup a simulation study in which we assume that the data generating model looks as
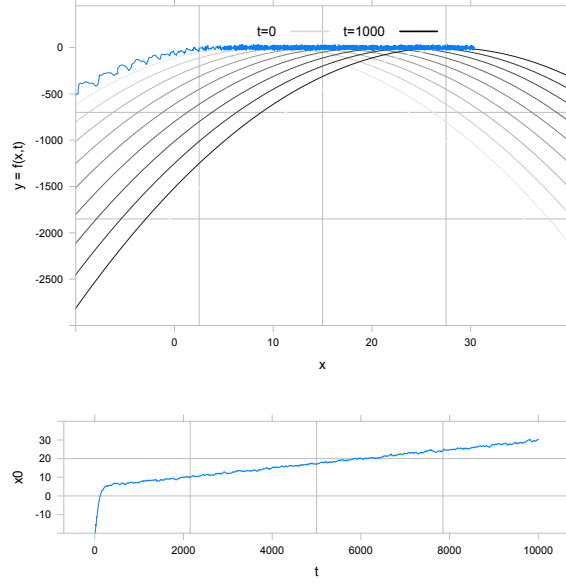
FIG 5. *Illustration of LiF in the case of concept drift. As the true maximum shifts (top panel) LiF is able to follow the maximum and keep x_0 close to x_{max} (bottom panel).*

follows:

$$y_t \sim \texttt{Bernoulli} \left(\frac{1}{1 + e^{-(10-x_t)}} \right) \qquad (15)$$

$$r_t = y_t x_t \qquad (16)$$

Intuitively the above specification indicates that the probability that a consumer chooses to buy a product decreases as the price, x, of the product increases, while the (expected) revenue is computed using the probability of a purchase given a specific price multiplied by that price. Given this setup the (expected) x_{max} is approximately 8.

Figure 6 shows the performance of LiF-II for two different starting values, $x_0 = 4$ and $x_0 = 15$, using the same set of tuning parameters as those used in Study 3 ($t = 10^4 = \mathcal{T}$, $A = 1$, $T = 100$, $\gamma = .1$). The only change in the algorithm compared to the earlier simulations is that $r_t = y_t x_i$ is integrated (summed) over instead of using the observed y_t directly. Also in this case, LiF finds x_{max} fairly quickly (in under 6000 iterations).

It has to be noted that too high starting values, and thereby a very low $Pr(y = 1|x_0)$ might lead to a failure to find x_{max} since LiF then get's stuck in a local "maximum": for very high values of x the revenue r will always be 0.
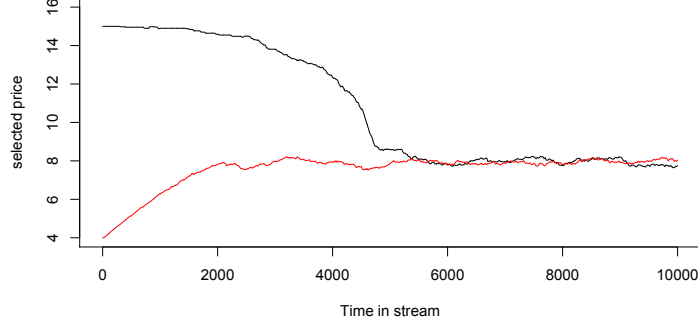
FIG 6. *Use of LiF to find the revenue maximizing sales-price for a firm: example of a setup in which the observed $y \in \{0, 1\}$.*

8. Simulation study 5: Empirical Regret

The previous studies show that LiF is effective in finding the value of x_{max}. However, the oscillation that is introduced clearly introduces search costs into the procedure: LiF continuously runs experiments with a certain amplitude in its variation in x to find x_{max}. In the previous simulations these search costs have not been considered, and hence while these simulations demonstrate that LiF finds the value of x_{max}, the previous simulation studies are uninformative regarding the costs of the procedure. To address this problem we run another simulation study in which we monitor the empirical *regret*

$$\mathcal{R}(t) = \sum_{i=1}^{t} (f(x_{max}) - (f(x_t))) \tag{17}$$

of the procedure. Thus, we compare over time in the data stream how much "is lost" when using LiF as compared to always selecting the exact right value of x that maximizes the outcome if the data generating process would have been known. We use the exact setup as used in Simulation Study 4 (exact same data generating model and tuning parameter settings), but we increase $\mathcal{T}$ to 10^5. Also, because of the noise and our interest in LiF as a general procedure, not merely in one specific attempt, we replicate the simulation $M = 100$ times.

To give insight in the performance of LiF when examining the regret of the procedure, we contrast the use of LiF not only to selecting the optimal value, but also to two other sequential experimentation scheme's:

- ϵ-*first:* in this approach we run a limited time (up to $n = 1000$) experiment in which we randomly sample values of x uniformly between 0 and 20. Subsequently, we fit a simple logistic regression modeling $Pr(y = 1|x) = \mathcal{L}(\beta_0 + \beta_1 x)$ where $\mathcal{L}()$ denotes the logit link (see also Equation 15), and

determine $x_\epsilon = \arg\max_x \mathcal{L}(\beta_0 + \beta_1 x)x$. The remaining $\mathcal{T} - n$ observations in the stream are allocated to x_ϵ.

- *Bootstrap Thompson Sampling (BTS)*: In this sequential experimentation scheme we again fit a simple logistic regression to estimate $Pr(y = 1|x)$. We use Stochastic Gradient Descent to update the parameters of the model at each time point during the data stream. Furthermore, we maintain $J = 100$ models each using an online half-sampling bootstrap to perform bootstrap Thompson sampling (See for details of this sequential allocation scheme Kaptein and Eckles, 2014). This gives J different estimates of the model parameters ($\{\beta_0^j, \beta_1^j\}$). We then randomly uniformly select j' out of $j = 1, \ldots, j = J$ and select treatment $x_{bts} = \arg\max_x \mathcal{L}(\beta_0^{j'} + \beta_1^{j'} x)x$. This bootstrapped sampling scheme quantifies the uncertainty in the model estimates and uses this directly to balance exploration (querying new values for x to learn more about the data-generating model), and exploitation (selecting the value of x which one believes leads to the highest outcomes).

Note that we choose random starting points of the parameter values for BTS that are relatively close to the true values, and that the functional form of the model that is used is the same as the true data generating model. Hence, this latter condition is expected to do very well on the current problem since it implements a lot of knowledge regarding the data-generating function that is not accessible to LiF.

Figure 7 shows the performance of LiF-II – in terms of average regret – compared to the ϵ-first and BTS. It is clear that the ϵ-first does not perform very well: logically, during the experimentation stage $t = 1, \ldots, t = n$ this method incurs a large regret. However, since the probability that the true x_{max} is found exactly in the experiment is smaller then 1, also after the experiment period (expected) linear regret is incurred. BTS performs much better in the long run: the regret is not linear but rather seems to be $\mathcal{O}(\sqrt{(t)})$, which is the proven minimal regret bound known for this problem (Agarwal et al., 2011).

Early on LiF performs very well on this problem; LiF is very efficient in finding x_{max}. It is even more efficient than BTS for small t, despite the fact that in the current setup BTS is heavily favored by using the correct form of the data generating model, something which is in practice very unlikely. However, in the long run the regret of LiF is lineair in t. This latter fact is easily explained: due to the continuous oscillations of x by adding $A\cos\omega t$ LiF keeps exploring the space and thus keeps incurring additional costs. Even if x_{max} has been found, these search costs are linear with t.

This simulation suggests that, in the bandit feedback case, LiF can be improved by gradually decreasing the amplitude of the oscillation: if A can be decreased as a function of (e.g.,) the approximated gradient as well as the current time in the stream, the exploration behavior of LiF can be systematically decreased over time in the stream. However, this would make LiF less sensitive to concept drift, which might in practice be infeasible. Hence, we currently regard the linear regret incurred by LiF as exploration costs necessary to ensure
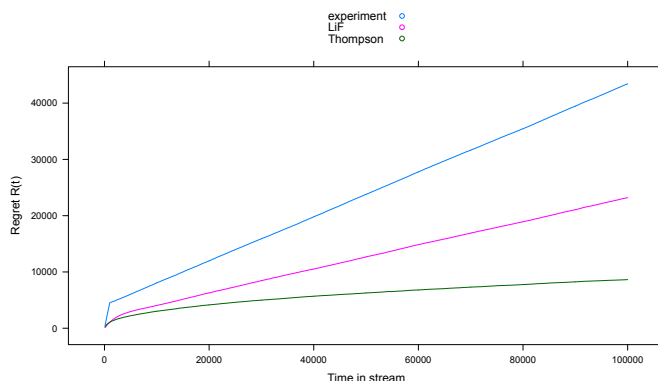
FIG 7. *Overview of the (mean) empirical regret of three possible sequential allocation schemes.*

its robustness in a changing environment.

9. Discussion and Future work

In this paper we presented Lock in Feedback as a method to find $\arg\max_x f(x)$ through sequential experiments. The method is appealing since it a) does not require the functional form of $f(x)$ to be known to derive its maximum, b) performs well in situations in which measurements are obtained with large noise, and c) allows following the maximum of a function even if that function changes over time. We have presented the basic mathematical arguments behind LiF, demonstrating how known (or imposed) oscillations in x can be used to determine the derivative(s) of $f(x)$ which can subsequently be used to find $\arg\max_x f(x)$. Next, we detailed two possible implementations of LiF and examined their performance for a variety of tuning parameter settings. We then showed that a streaming version of LiF is robust both to noise as well as concept drift.

We believe LiF can be of use in many sequential experimentation problems in which the independent variable is continuous; in the introduction we discussed pricing, medication dosing, and the selection of items by their difficulty as possible examples. LiF is extremely easy to implement, and very robust to noise and concept drift. We thus hope that LiF can be a valuable tool in treatment optimization in sequential experiments.

However, the current expose of LiF also introduces a number of questions. For example, the ability to use LiF for problems of higher dimensions, e.g., where $y = f(\vec{x})$ is a function of multiple variables, has not been explored here even though this extension relatively is easily made. Also, the suggested decrease of the amplitude in the bandit setting (Simulation Study 5) needs further scrutiny and begs for an analytical treatment of the use of LiF in stochastic optimization with bandit feedback (see, e.g., Agarwal et al., 2011). Finally, the currently proposed

version of LiF allows one to find local maxima (or minima), but convergence to a global maximum is not guaranteed. Throughout this paper, we have been considering unimodal functions, which might, in practical applications, be a too stringent assumption.

In this paper we have demonstrated the use of LiF only in cases where x is scalar. However, when x is a vector a very similar approach can be used to find the maximum of the function $f(\vec{x})$ in more than one dimension. In the two dimensional case LiF can be extended by oscillating both elements of x at different frequencies:

$$x_{1,t} = x_{1,0} + A_1 \cos \omega_1 t$$
$$x_{2,t} = x_{2,0} + A_2 \cos \omega_2 t$$

After oscillating both elements of x we observe $y_t = f(x_{1,t}, x_{2,t})$ and we can obtain information regarding the gradient by separately computing:

$$y_{1,\omega} = y_t \cos \omega_1 t$$
$$y_{2,\omega} = y_t \cos \omega_2 t$$

This simple extension allows for the use of LiF in higher dimensions. However, besides the fact that ω_1 and ω_2 should not be multiples of each other, the effects of the tuning parameters and the performance of this higher dimensional version of LiF need to be further examined.

Our proposed LiF algorithm, similar to many other procedures for function maximization, is prone to uncovering local maxima instead of global maxima. A logical solution to this problem would be to consider multiple starting points $\vec{x}_0$ which are oscillated independently (possibly alternating within a data stream). Effectively this would allow the experimenter to find multiple maxima. By evaluating the value of y one could decide on the best possible solution, or, one could pool the results of multiple alternating threats to update each of them. Such approaches, and their robustness to the existence of local maxima, needs further scrutiny.

References

Agarwal, A., Dekel, O., and Xiao, L. (2010). Optimal Algorithms for Online Convex Optimization with Multi-Point Bandit Feedback. In *COLT*, pages 28–40. Citeseer.

Agarwal, A., Foster, D. P., Hsu, D. J., Kakade, S. M., and Rakhlin, A. (2011). Stochastic convex optimization with bandit feedback. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 24*, pages 1035–1043. Curran Associates, Inc.

Allen, T. T., Yu, L., and Schmitz, J. (2003). An experimental design criterion for minimizing meta-model prediction errors applied to die casting process design. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 52(1):103–117.

Anagnostopoulos, C., Tasoulis, D. K., Adams, N. M., Pavlidis, N. G., and Hand, D. J. (2012). Online linear and quadratic discriminant analysis with adaptive forgetting for streaming classification. *Statistical Analysis and Data Mining*, 5(2):139–166.

Ansari, A. and Mela, C. F. (2003). E-Customization. *Journal of Marketing Research*, 40(2):131–145.

Audibert, J.-Y., Munos, R., and Szepesvári, C. (2009). Exploration-exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19):1876–1902.

Bardsley, W. G., Wood, R. M. W., and Melikhova, E. M. (1996). Optimal design: A computer program to study the best possible spacing of design points for model discrimination. *Computers & chemistry*, 20(2):145–157.

Berry, D. A. and Fristedt, B. (1985). *Bandit Problems: Sequential Allocation of Experiments*. Springer.

Beygelzimer, A., Langford, J., Reyzin, L., and Schapire, R. E. (2011). Contextual Bandit Algorithms with Supervised Learning Guarantees arXiv : 1002 . 4058v3 [cs . LG] 27 Oct 2011. 15.

Bubeck, S., Munos, R., and Stoltz, G. (2011a). Pure exploration in finitely-armed and continuous-armed bandits. *Theoretical Computer Science*, 412(19):1832–1852.

Bubeck, S., Munos, R., and Stoltz, G. (2011b). Pure exploration in finitely-armed and continuous-armed bandits. *Theoretical Computer Science*, 412(19):1832–1852.

Burnetas, A. N. and Katehakis, M. N. (1996). Optimal adaptive policies for sequential allocation problems. *Advances in Applied Mathematics*, 17(2):122–142.

Flaxman, A. D., Kalai, A. T., and McMahan, H. B. (2005). Online convex optimization in the bandit setting: gradient descent without a gradient. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 385–394. Society for Industrial and Applied Mathematics.

Gaber, M. M., Zaslavsky, A., and Krishnaswamy, S. (2005). Mining data streams. *ACM SIGMOD Record*, 34(2):18.

Jiang, Y., Shang, J., Kemerer, C. F., and Liu, Y. (2011). Optimizing E-tailer Profits and Customer Savings: Pricing Multistage Customized Online Bundles. *Marketing Science*, 30(4):737–752.

Kaptein, M. and Eckles, D. (2014). Thompson Sampling with the Online Bootstrap. *arXiv preprint arXiv:1410.4009*.

Kuck, H., de Freitas, N., and Doucet, A. (2006). SMC samplers for Bayesian optimal nonlinear design. In *Nonlinear Statistical Signal Processing Workshop, 2006 IEEE*, pages 99–102. IEEE.

Kung, M., Monroe, K. B., and Cox, J. L. (2002). Pricing on the Internet. *Journal of Product & Brand Management*, 11(2):274–288.

Lai, T. L. (1987). Adaptive Treatment Allocation and the Multi-Armed Bandit Problem. *The Annals of Statistics*, 15(3):1091–1114.

Li, L., Chu, W., and Langford, J. (2010). An Unbiased, Data-Driven, Offline Evaluation Method of Contextual Bandit Algorithms. 26:13.

Marschner, I. C. (2007). Optimal design of clinical trials comparing several treatments with a control. *Pharmaceutical statistics*, 6(1):23–33.

McClelland, G. H. (1997). Optimal design in psychological research. *Psychological Methods*, 2(1):3–19.

Myung, J. and Pitt, M. (2009a). Bayesian adaptive optimal design of psychology experiments. *Proceedings of the 2nd International Workshop in Sequential Methodologies (IWSM2009)*, pages 1–6.

Myung, J. I., Cavagnaro, D. R., and Pitt, M. A. (2013). A tutorial on adaptive design optimization. *Journal of mathematical psychology*, 57(3):53–67.

Myung, J. I. and Pitt, M. A. (2009b). Optimal experimental design for model discrimination. *Psychological review*, 116(3):499.

O'Brien, T. E. and Funk, G. M. (2003). A gentle introduction to optimal design for regression models. *The American Statistician*, 57(4):265–267.

Poggio, T., Voinea, S., and Rosasco, L. (2011). Online Learning, Stability, and Stochastic Gradient Descent. *Artificial Intelligence*, 8(11):11.

Sapareto, S. A. and Dewey, W. C. (1984). Thermal dose determination in cancer therapy. *International Journal of Radiation Oncology*Biology*Physics*, 10(6):787–800.

Scofield, J. H. (1994). Frequency-domain description of a lock-in amplifier. *American Journal of Physics*, 62(2):129.

Scott, S. L. (2010). A modern Bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658.

Shamir, O. (2012). On the complexity of bandit and derivative-free stochastic convex optimization. *arXiv preprint arXiv:1209.2388*.

Whittle, P. (1980). Multi-armed bandits and the Gittins index. *Journal of the Royal Statistical Society Series B Methodological*, 42(2):143–149.

Yue, Y., Hong, S., and Guestrin, C. (2012). Hierarchical exploration for accelerating contextual bandits. *arXiv preprint arXiv:1206.6454*.

Zhang, S. and Lee, M. D. (2010). Optimal experimental design for a class of bandit problems. *Journal of Mathematical Psychology*, 54(6):499–508.

Algorithm for finding the exact maximum of a parabola using the second order approximation.

Let's suppose that the curve $y = f(x)$ is a parabola:

$$y = -\alpha(x - x_0)^2 + \gamma$$

Clearly, $f(x)$ has a maximum for $x = x_0$. Furthermore, the second derivative is always equal to -2α, regardless the value of x. Interestingly, the value of α can be easily extracted from the data accumulated during the lock-in procedure. For this purpose, $y(t)$ has to be multiplied by $\cos{(2\omega t)}$. Following the steps illustrated in eq. 5, eq. 6, and eq. 10, one obtains:

$$y_{2\omega} = \frac{TA^2}{8} \left(\frac{\partial^2 f}{\partial x^2} \bigg|_{x=x_0} \right)$$
$$+ \int_0^T \epsilon \cos{(2\omega t)}$$

which allows us to calculate α as:

$$\alpha = \frac{4y_{2\omega}}{TA^2}$$